



nextwork.org

Set Up Kubernetes Deployment



saqibh49@gmail.com

```
aws | ⚡ | Search [Option+S] ⓘ | United States (Ohio) | Saqib Hossain (S104-8260-3806) ▾ | saqib-JAM-Admin
```

```
[ec2-user@ip-172-31-26-211 ~]$ cd nextwork.flask.backend
[ec2-user@ip-172-31-26-211 nextwork.flask.backend: No such file or directory
[ec2-user@ip-172-31-26-211 ~]$ ls
nextwork-flask-backend
[ec2-user@ip-172-31-26-211 ~]$ cd nextwork-flask-backend
[ec2-user@ip-172-31-26-211 nextwork-flask-backend]$ ls
Dockerfile requirements.txt requirements.txt
[ec2-user@ip-172-31-26-211 nextwork-flask-backend]$ docker build -t nextwork-flask-backend
[+] Building 11.1s (10/10) FINISHED
docker:default
[+] 1/10: Building 11.1s
[+] 1/10: 100% complete
[+] 1/10: load build definition from Dockerfile
[+] 1/10: transferring Dockerfile... 269B
[+] 1/10: 100% complete
[+] 1/10: for Dockerfile: Dockerfile
[+] 1/10: 100% complete
[+] 1/10: load .dockerignore
[+] 1/10: 100% complete
[+] 1/10: transferring context: 2B
[+] 1/10: 100% complete
[+] 1/10: FROM docker.io/library/python:3.9-alpine
[+] 1/10: 100% complete
[+] 1/10: RUN pip install -r requirements.txt
[+] 1/10: 100% complete
[+] 1/10: sha256:a381ce006821bdc2a0d62e61090f912e9f23d9e1f61007e58328863bd7708 1.73kB / 1.73kB
[+] 1/10: sha256:ac5feec2244c78ecbb822308adff0875572ffdb2124470d 5.19kB / 5.19kB
[+] 1/10: sha256:2d35ebdb57d9971fea0ca1582a79935adf8058b2c32db163c98822e5dfa1b 3.80MB / 3.80MB
[+] 1/10: sha256:27711585a56e347c5206607e7f177b1a01601661f6291347a3d04d10c408b / 15.40MB
[+] 1/10: sha256:c99be6b43b3acd4750dbd3deeb2228d5e5a9a99deea7218ce3deda7f2ca039c 10.29kB / 10.29kB
[+] 1/10: sha256:2d35ebdb57d9971fea0ca1582a79935adf8058b2c32db163c98822e5dfa1b
[+] 1/10: sha256:0692fa0d91a130440d4247d7aa253ef05e86699f1ce974ceccabeb1b1 4.77kB / 4.77kB
[+] 1/10: sha256:181081193159a568cd76752066b7f5d5137754eb3d501cf866ff9a29313d7c3db806d 0.98
[+] 1/10: sha256:27711589a568cd76752066b7f5d5137754eb3d501cf866ff9a29313d7c3db806d
[+] 1/10: sha256:9806928fa0d94a3d2440c42430d7caa253eeff05e8669881cee924ceccabeb1b3 0.08
[+] 1/10: internal: load build context
[+] 1/10: 100% complete
[+] 1/10: load context: 42.30kB
[+] 1/10: WORKDIR /app
[+] 1/10: COPY requirements.txt requirements.txt
[+] 1/10: RUN pip3 install -r requirements.txt
[+] 1/10: COPY .
[+] 1/10: 100% complete
[+] 1/10: exporting layers
[+] 1/10: writing image sha256:5972d8be97e4ff46591563dca5dc5fe250edf7ea424b32b548abcc72b77e66f4
[+] 1/10: naming to docker.io/library/nextwork-flask-backend
[ec2-user@ip-172-31-26-211 nextwork-flask-backend]$
```

i-001366ec9a42e8673 (nextwork-eks-instance)

PublicIPs: 13.58.99.178 PrivateIPs: 172.31.26.211

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

saqibh49@gmail.com

NextWork Student

nextwork.org

Introducing Today's Project!

In this project, I will clone a backend app from GitHub, build a Docker image of it, and push that image to Amazon ECR. I'm doing this project because containerizing applications and storing them in a registry is a key part of deploying apps on Kubernetes, and I need to understand this workflow if I want to work with production environments.

Tools and concepts

I used Amazon EKS, EC2, IAM, CloudFormation, Git, Docker, and Amazon ECR to containerize a backend application and prepare it for Kubernetes deployment. Key steps include launching an EC2 instance, setting up eksctl, creating an EKS cluster, cloning the backend code from GitHub, building a Docker image, troubleshooting permission errors, and pushing the image to ECR for my cluster to use.

Project reflection

This project took me approximately 2 hours. The most challenging part was solving for all the errors when setting up docker. My favourite part was seeing the backend and understanding how everything comes together to create a flask app.

saqibh49@gmail.com

NextWork Student

nextwork.org

Something new that I learnt from this experience was how many tools work together just to get an app ready for Kubernetes. It's not just about writing code — you need Docker to package it, ECR to store it, EKS to run it, and IAM to make sure everything has the right permissions along the way.

saqibh49@gmail.com

NextWork Student

nextwork.org

What I'm deploying

To set up today's project, I launched a Kubernetes cluster. Steps I took to do this included launching an EC2 instance, connecting to it, installing eksctl, attaching an IAM role with the right permissions, and running the eksctl create cluster command to spin up my EKS cluster with its node group.

I'm deploying an app's backend

Next, I retrieved the backend that I plan to deploy. An app's backend means the server-side code that handles the logic, data processing, and database interactions that users don't see directly. I retrieved the backend code by installing Git on my EC2 instance using `sudo dnf install git -y`, then running `git clone` to download the `nextwork-flask-backend` repository from GitHub directly onto my instance.



saqibh49@gmail.com

NextWork Student

nextwork.org

```
aws [■■■] Q Search [Option+S] ⓘ United States (Ohio) ▾ Saqib Hossain (S104-8260-3803) ▾ saqib-JAM-Admin
[ec2-user@ip-172-31-26-211 ~]$ sudo dnf update
[sudo] password for saqib: 
Last metadata expiration check: 0:00:02 ago on Sat Feb 21 13:21:09 2026.
Dependencies resolved.
Nothing to do.
Complete!
List of metadata expiration check: 0:00:02 ago on Sat Feb 21 13:21:09 2026.
Dependencies resolved.

=====
Packages
=====
Architecture Version Repository Size
=====
Installing:
git           x86_64   2.50.1-1.amzn2023.0.1      amazonlinux      53 kB
Installing dependencies:
git-core       x86_64   2.50.1-1.amzn2023.0.1      amazonlinux      4.9 MB
git-doc        noarch   1.19.2-1.amzn2023.0.1     amazonlinux      21 kB
perl-error     noarch   1.10.17029-5.amzn2023.0.2    amazonlinux      41 kB
perl-LFile-Find
perl-MIME-Header
perl-TermReadKey x86_64   2.38.9-9.amzn2023.0.2      amazonlinux      41 kB
perl-TermReadKey x86_64   2.38.9-9.amzn2023.0.2      amazonlinux      36 kB
perl-1-lib      x86_64   0.65-477.amzn2023.0.7      amazonlinux      15 kB

Transaction Summary
=====
Install 8 Packages

Total download size: 7.9 M
Installed size: 41 M
Dependencies resolved:
(1/8): git-2.50.1-1.amzn2023.0.1.x86_64.rpm          1.4 MB/s | 53 kB  00:00
(2/8): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm  44 MB/s | 2.8 MB  00:00
(3/8): perl-MIME-Header-2.38.9-9.amzn2023.0.2.noarch.rpm 113 kB/s | 4.9 MB  00:00
(4/8): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm      52 MB/s | 4.9 MB  00:00
(5/8): perl-LFile-Find-1.37-477.amzn2023.0.7.noarch.rpm 826 kB/s | 25 kB   00:00
(6/8): perl-1-lib-0.65-477.amzn2023.0.7.x86_64.rpm   1.4 MB/s | 1.1 MB  00:00
(7/8): perl-TermReadKey-2.38.9.amzn2023.0.2.x86_64.rpm 1.5 MB/s | 36 kB   00:00
(8/8): perl-TermReadKey-2.38.9.amzn2023.0.2.x86_64.rpm  533 kB/s | 15 kB   00:00

Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing transaction
  Installing : git-core-2.50.1-1.amzn2023.0.1.x86_64
  Installing : git-2.50.1-1.amzn2023.0.1.noarch
  Installing : perl-1-lib-0.65-477.amzn2023.0.7.x86_64
  Installing : perl-TermReadKey-2.38.9.amzn2023.0.2.x86_64
1/8
2/8
3/8
4/8

CloudShell Feedback ⓘ Console Mobile App
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

saqibh49@gmail.com

NextWork Student

nextwork.org

Building a container image

Once I cloned the backend code, my next step is to build a container image of the backend. This is because a container image packages up the app and everything it needs to run into one unit, making it easy to deploy consistently across any environment — especially on my Kubernetes cluster where it'll actually be running.

When I tried to build a Docker image of the backend, I ran into a permissions error because my user didn't have permission to access Docker. Basically, Docker needs root-level access to run, and my ec2-user wasn't part of the Docker group yet, so it got denied.

To solve the permissions error, I added ec2-user to the docker group. The Docker group is basically a list of users that are allowed to communicate with Docker without needing root access. Once I added my user to the group, I could run Docker commands without getting permission denied.



saqibh49@gmail.com

NextWork Student

nextwork.org

saqibh49@gmail.com

NextWork Student

nextwork.org

Container Registry

I'm using Amazon ECR in this project to store my Docker image so my Kubernetes cluster can pull it when it needs to run the app. ECR is a good choice for the job because it's fully managed by AWS, integrates seamlessly with EKS, and keeps my images private and secure without me having to set up my own registry.

Container registries like Amazon ECR are great for Kubernetes deployment because they give your cluster a central place to pull images from. Instead of manually copying images to each node, Kubernetes just pulls the image from the registry whenever it needs to spin up a new container — making deployments faster, more consistent, and easier to manage.

The screenshot shows a terminal window titled "aws" with the command line interface. The user is performing the following steps:

- Using `aws ecr get-login-password` to retrieve the password for the ECR repository.
- Logging into the ECR repository using `docker login` with the provided credentials.
- Pushing the Docker image to the repository using `docker push`.
- Tagging the image with a specific version using `docker tag`.
- Verifying the pushed image with `docker images`.

The terminal output includes error messages related to repository names and tags, and ends with a success message indicating the image was pushed successfully.

saqibh49@gmail.com

NextWork Student

nextwork.org

EXTRA: Backend Explained

After reviewing the app's backend code, I've learnt that this is a pretty straightforward API that pulls data from an external source and reformats it for the frontend. It helped me understand how Flask apps are structured, how APIs call other APIs, and how the backend handles things like filtering data and setting up CORS — all stuff I've dealt with in previous projects but now I can see it directly in the code.

Unpacking three key backend files

The requirements.txt file lists all the Python libraries the app needs to run — like Flask for the web framework, flask-restx for building the API, requests for making HTTP calls, and werkzeug for handling the underlying server utilities. When the Docker image is built, it reads this file and installs everything automatically so the app has all its dependencies ready to go.

The Dockerfile gives Docker instructions on how to build the container image step by step. Key commands in this Dockerfile include FROM, which sets the base image to a lightweight Python environment, WORKDIR, which sets the working directory inside the container, COPY, which moves the app files into the container, RUN, which installs all the dependencies from requirements.txt, and CMD, which tells the container to run the app when it starts up.

saqibh49@gmail.com

NextWork Student

nextwork.org

The app.py file contains the actual backend application code. It sets up a Flask API with one endpoint — /contents/<topic> — that takes a topic, searches the Hacker News API for articles matching that topic, and returns a list of results with each article's ID, title, and URL. It also includes CORS headers so any frontend can make requests to it, and runs the app on port 8080.



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

